Computational Learning Theory
Trinity Term 2021
Week 1

Problem Sheet 3

Instructions: The problem sheets are designed to increase your understanding of the material taught in the lectures, as well as to prepare you for the final exam. You should attempt to solve the problems on your own after reading the lecture notes and other posted material, where applicable. Problems marked with an asterisk are optional. Once you have given sufficient thought to a problem, if you are stuck, you are encouraged to discuss with others in the course and with the lecturer during office hours. You are *not permitted* to search for solutions online.

1 Weak Learning CONJUNCTIONS and PARITIES

Consider the instance space $X_n = \{0,1\}^n$. Consider the following hypothesis class:

$$H_n = \{0, 1, x_1, \bar{x}_1, x_2, \bar{x}_2, \dots, x_n, \bar{x}_n\}.$$

The hypothesis class contains $2n + 2$ functions. The functions "0" and "1" are constant and predict 0 and 1 on all instances in X_n. The function "x_i" evaluates to 1 on any $a \in \{0,1\}^n$ satisfying $a_i = 1$ and 0 otherwise. Likewise, the function "$\bar{x}_i$" evaluates to 1 on any $a \in \{0,1\}^n$ satisfying $a_i = 0$ and 0 otherwise. Thus a single bit of the input determines the value of these functions; for this reason these functions are sometimes referred to as *dictator* functions.

1. Show that the class CONJUNCTIONS is $\frac{1}{10n}$-weak learnable using H.

   Hint: The factor 10 is not particularly important, just a sufficiently large constant.

2. Let $\mathsf{CONJUNCTIONS}_k$ denote the class of conjunctions on at most k literals. Give an algorithm that PAC-learns $\mathsf{CONJUNCTIONS}_k$ and has sample complexity polynomial in k, $\log n$, $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$. What would be the sample complexity if you had used the algorithm for learning CONJUNCTIONS discussed in the lectures?

   Hint: First show that the weak learning algorithm in the previous part can be modified to be a $\frac{1}{10k}$-weak learner in this case.

3. Show that there does not exist a weak learning algorithm for PARITIES using H.

2 Teaching Dimension

We consider a new model of learning with the aid of a teacher. In this model, instead of receiving examples randomly from a distribution, a teacher can provide examples that are most *helpful* to a learner. Let X be a finite instance space. Given a concept class C and a target concept $c \in C$, we say that a sequence T of labelled examples is a *teaching sequence* for $c \in C$, if c is the only concept in C which is consistent with T. Let $T(c)$ be the set of all teaching sequences for $c \in C$. The *teaching dimension* of concept class C is then defined to be,

$$\mathsf{TD}(C) = \max_{c \in C} \min_{T \in T(c)} |T|$$

where $|T|$ denotes the number of examples in the sequence T.

1. Give an example of a concept class C for which $\mathsf{TD}(C) > \mathsf{VCD}(C)$.
2. Give an example of a concept class C for which $\mathsf{TD}(C) < \mathsf{VCD}(C)$.
3. Show that for any concept class C, $\mathsf{TD}(C) \leq |C| - 1$.
4. Show that for any concept class C, $\mathsf{TD}(C) \leq \mathsf{VCD}(C) + |C| - 2^{\mathsf{VCD}(C)}$.

3 Learning MONOTONE-DNF is equivalent to learning DNF

The class $\mathsf{MONOTONE\text{-}DNF}_{n,s}$ over $\{0,1\}^n$ contains boolean functions that can be represented as DNF formulae with at most s terms over n variables, and where each term only contains positive literals. Then define,

$$\mathsf{MONOTONE\text{-}DNF} = \bigcup_{n \geq 1} \bigcup_{s \geq 1} \mathsf{MONOTONE\text{-}DNF}_{n,s}.$$

The class DNF is defined analogously, except that the literals in the terms may also be negative. An efficient learning algorithm is allowed time polynomial in n, s, $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$. Show that if the class MONOTONE-DNF is efficiently PAC-learnable, then so is DNF.

Remark: We have shown in the lectures that MONOTONE-DNF is efficiently exactly learnable using membership and equivalence queries, and hence also efficiently PAC-learnable using membership queries (see Problem 4). On the other hand, there is no known algorithm for PAC-learning DNF even when membership queries are allowed. In fact, under a suitable crypotographic assumption, it has been shown that PAC-learning DNF with or without membership queries is equivalent (Angluin and Kharitonov, 1991).

4 From Exact Learning to PAC Learning with Membership Queries

Let C be a concept class that is exactly efficiently learnable using membership and equivalence queries. We will consider the learnability of C in the standard PAC framework. Prove that if in addition to access to the example oracle, $\mathsf{EX}(c, D)$, the learning algorithm is allowed to make membership queries, then C is *efficiently* PAC-learnable. Formally, show that there exists a learning algorithm that for all $n \geq 1$, $c \in C_n$, D over X_n, $0 < \epsilon < 1/2$ and $0 < \delta < 1/2$, that with access to the oracle $\mathsf{EX}(c, D)$ and the membership oracle for c and with inputs ϵ, δ and size(c), outputs h that with probability at least $1 - \delta$ satisfies $\mathrm{err}(h) \leq \epsilon$. The running time of L should be polynomial in n, size(c), $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$ and the h should be from a hypothesis class H that is polynomially evaluatable.

5 Learning Rectangles using Statistical Queries

We will consider an extension of the statistical query model, where in addition to making queries of the form (χ, τ) to the oracle $\mathsf{STAT}(c, D)$, the learning algorithm is allowed access to *unlabelled* examples from D, *i.e.,* it may get points $x \in X$ drawn according to D, but not the labels $c(x)$.

1. Briefly argue why any concept that is (efficiently) learnable with access to $\mathsf{STAT}(c, D)$ and unlabelled examples, is also (efficiently) learnable with access to the noisy example oracle, $\mathsf{EX}^{\eta}(c, D)$.

2. Give an efficient algorithm for learning rectangles in the plane using $\mathsf{STAT}(c, D)$ and unlabelled examples.

References

Dana Angluin and Michael Kharitonov. When won't membership queries help? In *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 444–454. ACM, 1991.